THE GENERATION OF RANDOM NUMBERS FROM VARIOUS PROBABILITY DISTRIBUTIONS

JOHN E. HOWE

THE GENERATION OF RANDOM NUMBERS

FROM VARIOUS PROBABILITY DISTRIBUTIONS

* * * * * *

John E. Howe

THE GENERATION OF RANDOM NUMBERS

FROM VARIOUS PROBABILITY DISTRIBUTIONS

by

John E. Howe

Lieutenant, United States Navy

Submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE
IN
OPERATIONS RESEARCH

United States Naval Postgraduate School
Monterey, California

1 9 6 5

THE GENERATION OF RANDOM NUMBERS

FROM VARIOUS PROBABILITY DISTRIBUTIONS

by

John E. Howe

This work is accepted as fulfilling

the thesis requirements for the degree of

MASTER OF SCIENCE

IN

OPERATIONS RESEARCH

from the

United States Naval Postgraduate School

ABSTRACT

Methods are developed, and Fortran 63 CODAP computer programs are demonstrated, to generate random numbers from the uniform, normal (including multivariate normal), Poisson, and exponential probability distributions. Various statistical tests are described and the results of the application of these tests to the generators are tabulated. A general method for generating random numbers from a large class of distributions is described. The methods of generation are optimized to provide an accurate generator while producing numbers at a maximum rate. The uniform generator that is used as a basis for the other generators is of the congruential type and is capable of generating 1800 numbers per second.

TABLE OF CONTENTS

1. Introduction.

The need for a rapid reliable source of random numbers from a prescribed distribution is well recognized, and is likely to become even more pressing in military circles in view of the Department of Defense's increased interest in Operations Analysis. This thesis presents methods and programs, some old, some new, for generating random numbers from various specified distributions. Some statistical tests of the programs and their results are described.

All methods assume a uniform random number generator is available. A thesis by Barron [1] has a good bibliography on this subject. The generator used here, and described more completely in Section 3, is of the mixed congruential type. While some uniform generators may have advantages over the one used, this one seems to perform very well, at the same time as being as fast as any demonstrated. Since this generator is used as a basis for all the others it should be remembered that no generator can be considered 'perfect', especially in the continuous distribution case, since the computer is limited to a finite set of possible numbers. However, for practical purposes this inaccuracy is not important. Other sources of inaccuracy can be important, however. The numbers generated must have the property of randomness, and must faithfully represent the desired distribution. These properties are measured by testing samples of the generated numbers. Some of the methods of generating numbers in this paper theoretically provide an exact

transformation from the uniform distribution to the desired distribution. An example of this is the half-Gaussian method of generating normal random numbers[1]. If we assume that the uniform numbers are accurate then we are led to the conclusion that the normal numbers are also accurate and that tests of these numbers would be superfluous. However, tests are performed to assure that the uniform numbers were so good that they did not bias the derived distribution. Other techniques such as Marsaglia's technique (see Section 4) for generating normal random numbers are in a sense curve-fitting techniques and only provide a controllably good approximation to the real distribution. The advantage of the approximation techniques is in the far greater speed with which they may provide the desired numbers. The user who needs to draw numbers from a distribution he suspects is normal, with inaccurately measured mean and hypothesized variance, is not in need of a generator that is accurate in the sixth decimal place, however, he still would like to be assured that having assumed a distribution and its parameters he will be able to generate numbers with the appropriate shape and with good properties of randomness.

[1]The phrase 'normal random numbers' and other like phrases should be read as 'numbers distributed as if coming from the normal distribution.'

2. Testing the Generators.

2.1 General Discussion.

The uniform generator chosen here has been tested extensively by others. The tests that have been applied include frequency tests, serial tests, moment tests, poker tests, gap tests, and many others. As mentioned in the introduction, this generator is as good as can be found, considering the requirement for speed. However, the derived distributions will be tested to overcome any doubts there may be about the transformation. The numbers will be tested mainly to measure the faithfulness with which they represent the derived distribution. The randomness is provided by the uniform generator. If the randomness is not satisfactory then the uniform generator must be blamed, not the transformation. A slower but more satisfactory method is available if the need is felt.[1] Thus the tests used here, the moment test, the hypothesis tests on the mean and variance, and the Kolmogorov-Smirnov goodness of fit test, are not designed to detect special types of non-randomness, such as is detected by the poker test and other similar tests.

Martin Greenberger has written an interesting article [17] on this subject. He presents the results of an investigation by Joseph Lach [18] at Yale University in which Lach showed that the congruential method of generating uniform random numbers has a

[1] See page 22.

predictable non-randomness in second order serial correlation. The lesson is clear. Having found an undersirable feature in a generator, it is generally possible to modify the generator to eliminate the feature, however, we can be sure that we have introduced another aberration of some kind, even though its form may be hard to determine. When the user asks, "Is this generator good enough?", the obvious retort is "good enough for what?" No one generator is suitable for all applications, but the generator used here will be good enough for most. If the user thinks that this is not so in his application, he has at his resources the modifying methods of Marsaglia [8] or Lach with the penalty of longer generation times.

2.2 Moment tests

The first four moments are calculated and are compared with the theoretical moments for each of the distributions. A more appealing statistic than the sample moment may be the unbiased estimator of the moment, however for large sample sizes such as are used here, this varies very little from the sample moment, and the sample moment is easier to handle in other uses- such as hypothesis testing. The unbiased estimator of the variance is:

$$\hat{\sigma}^2 = \frac{1}{N-1} \sum (X_i - \bar{X})^2 \quad [1]$$

The statistic used here is the sample second moment:

$$M2 = \frac{1}{N} \sum (X_i - \bar{X})^2$$

[1]All numerations are on the index 'i' which runs from 1 to N, the sample size, unless otherwise indicated.

The first moment is the mean:

$$\bar{X} = M1 = \frac{1}{N}\sum X_i$$

The third and fourth moments are:

$$M3 = \frac{1}{N}\Big[\left(\sum X_i^3\right) - \frac{3}{N}\left(\sum X_i\right)\left(\sum X_i^2\right) + \frac{2}{N}\left(\sum X_i\right)$$
$$\left(\sum X_i\right)^3\Big]$$

$$M4 = \frac{1}{N}\Big[\left(\sum X_i^4\right) - \frac{4}{N}\left(\sum X_i\right)\left(\sum X_i^3\right) + \frac{6}{N^2}\left(\sum X_i\right)^2$$
$$\left(\sum X_i^2\right) - \frac{3}{N^3}\left(\sum X_i\right)^4\Big]$$

2.3 Hypothesis tests on the mean and variance.

The availability of large sample sizes is used in designing tests on the mean and variance. The cental limit theorem is used where possible to simplify the test procedures.

2.3.1 Test on the mean.

This test is applied to the normal distribution. The hypothesis is that the mean is zero; the alternate hypothesis is that the mean is not zero. The test is performed by calculating the statistic Y, where

$$Y = \sqrt{N}\,\bar{X}$$

The decision rule at the alpha level becomes:

Accept the hypothesis when $-K_{\alpha/2} < Y < K_{\alpha/2}$

At the 10% level this rule becomes:

Accept the hypothesis when $-1.645 < Y < 1.645$

The justifications for using this test in preference to the 'T' test are that the variance can be assumed to be one, and that a large sample size is being used.

2.3.2 Test on the variance.

Again for the normal distribution, the hypothesis is that

the variance is one; the alternate hypothesis is that the variance is not one. The test is performed by calculating the statistic Z, where

$$Z = \frac{S-(N-1)}{\sqrt{2(N-1)}}$$

$$S = \sum (X_i - \bar{X})^2$$

The decision rule at the 10% level becomes:

Accept the hypothesis when $-1.645 < Z < 1.645$.

2.4 The Kolmogorov-Smirnov goodness of fit test.

In Massey's discussion of this test [6] , he presents evidence to indicate that this test may in many circumstances be better than the more usual chi-squared goodness of fit test. To test the hypothesis that a sample of size N comes from theorized distribution, the cumulative step function $S_N(x)$ is formed.

$$S_N(x)=k/n$$

where k is the number of observations less than or equal to x. The selection of x is arbitrary within certain limits. In this paper x is chosen so that there are either twenty or fifty equal intervals spanning the sample space. $S_N(x)$ is compared with the theoretical value of the cumulative distribution, F(x). The maximum difference d is calculated.

$$d=\max|F(x)-S_N(x)|$$

Tables due to Smirnov [7] give certain critical points of the distribution for various sample sizes. For sample sizes over 35, and at the 10% level of significance, if

$$d/n < 1.22/\sqrt{N}$$

the sampled distribution is accepted as the hypothesized distribution.

2.5 The chi-squared goodness of fit test.

This test was developed by Pearson and represents the earliest non-parametric decision-making test in statistics. Partly because of the length of time the test has been in existence, many drawbacks to the test have been noted, however it still stands as a useful and much used test. For this test the sample space has been divided into k intervals and the number of sample observations in each interval is noted.

$$X^2 = \sum_{i=1}^{k} \frac{(q_i - m_i)^2}{m_i}$$

where q_i is the observed number of sample observations in each interval, and m_i is the expected number. Pearson showed that X^2 has the chi-squared distribution with k-1 degrees of freedom. The decision rule at the alpha level becomes: if $X^2 \leq \chi^2_{k-1}(\alpha)$ accept the hypothesis that the distribtuion is as postulated. There are several problems associated with the application of this test. How should the interval size be determined? How many intervals should there be? Mann and Wald [14] studied this problem and formulated a criterion for the selection of k. Williams [15] notes that this criterion is not particularly sensitive to even a reduction by a factor of two in the number of intervals. The use of equal probability intervals vice equal length intervals is also recommended. However, the basis for this recommendation is unclear. It is agreed that very low probability intervals such as would occur

in the tails in an equal interval length division of the normal density function should be avoided. The test is applied here using equal length intervals. Low probability intervals are avoided by 'pooling' several intervals until the probability is of the same order of magnitude as in the other intervals.

2.6 Scatter diagrams

The best type of test to apply to the generator initially is some type of scatter diagram. The scatter diagram can often immediately give an intuitive idea as to whether the generator is behaving properly. In fact the scatter diagram can be a very powerful tool for rejecting a generator- more sophisticated techniques are needed to accept the generator, however. The scatter diagram that was used here was constructed by plotting the first number generated versus the second, the third versus the fourth, and so on. This type of plot will also enable us to look for correlations similar to those found by Lach [18] and discussed further in Section 2.1.

3. The Uniform Distribution.

3.1 Disbribution characteristics.

It is desired to generate numbers such that:

$$f_X(x) = \begin{cases} 0 & x \le 0 \\ 1 & 0 < x < 1 \\ 0 & x \ge 1 \end{cases}$$

The first four moments are: M1 = 1/2 : M2 = 1/12 ; M3 = 0 ; M4 = 1/80. The uniform distribution is usually specified in terms of its interval - uniform on the interval from zero to one (denoted here by U (0,1)). In the general case U (A,B), a simple transformation from U (0,1) is:

URAB = (UR01)(B-A) + A

where UR01 is the number provided by the U (0,1) generator, and URAB is the random number uniform on the interval (A,B).

3.2 Methods of generation.

3.2.1 Many techniques have been used over the years. For some particular applications such a method as table look-up may be suitable. However for our purposes what is desired is a rapid, 'accurate' method for the computer to produce a practically inexhaustible supply of numbers.

3.2.2 An early computational scheme was called the mid-square method. In this procedure two starting values, say A1 and A2, are multiplied together; the middle set of bits (usually 24) are extracted as the third random number A3; then A2 and A3 are multiplied together and the algorithm is continued in a similar fashion. This method has performed well in many tests but

unfortunately degenerates to all zeros in a relatively short time.

3.2.3 An improved computational scheme was tested extensively by Hull and Dobell [3] . This method forms a series of numbers A_i, where

$$A_{R+1} = BA_R + C \qquad \text{modulus } 2^{47}$$

B and C are constants to be selected.

3.2.4 A special form of this generator where C = 0 is the generally recommended form. C is set equal to zero because it does not improve the characteristics of the generator and it adds to computation time. The selection of B and the starting number X has been the subject of considerable study. Barron and others have shown that:

$$X_{R+1} = (2^{19} + 3)X_R \quad \text{modulus } 2^{47}$$

where X is either 1, or 2^{48} -1, or any number naturally generated in the sequence, is an excellent generator. Since this generator had been tested extensively previously no attempt was made to test it rigorsly although several interesting characteristics were noted. Some of these are included here. The generator was run down through the first 10^7 to 10^8 numbers. A number at the end of this sequence was extracted for possible alternate use as a starting number. It can be found in Appendix I. A graph of the mean of the first 10000 times i numbers, where i runs from 1 to 100 is plotted against the index i. This plot is compared with curves of = $K_{.05}/$ 10000i + 0.50000 versus i. The more our data plots between these curves the better.

We would expect it to be between the curves 90% of the time. As the following graph shows our generator does not quite live up to this expectation. A scatter diagram of the type suggested by Lach [18] was plotted. The graph on page 13 consists of 3000 points constructed as described in Section 2.6.

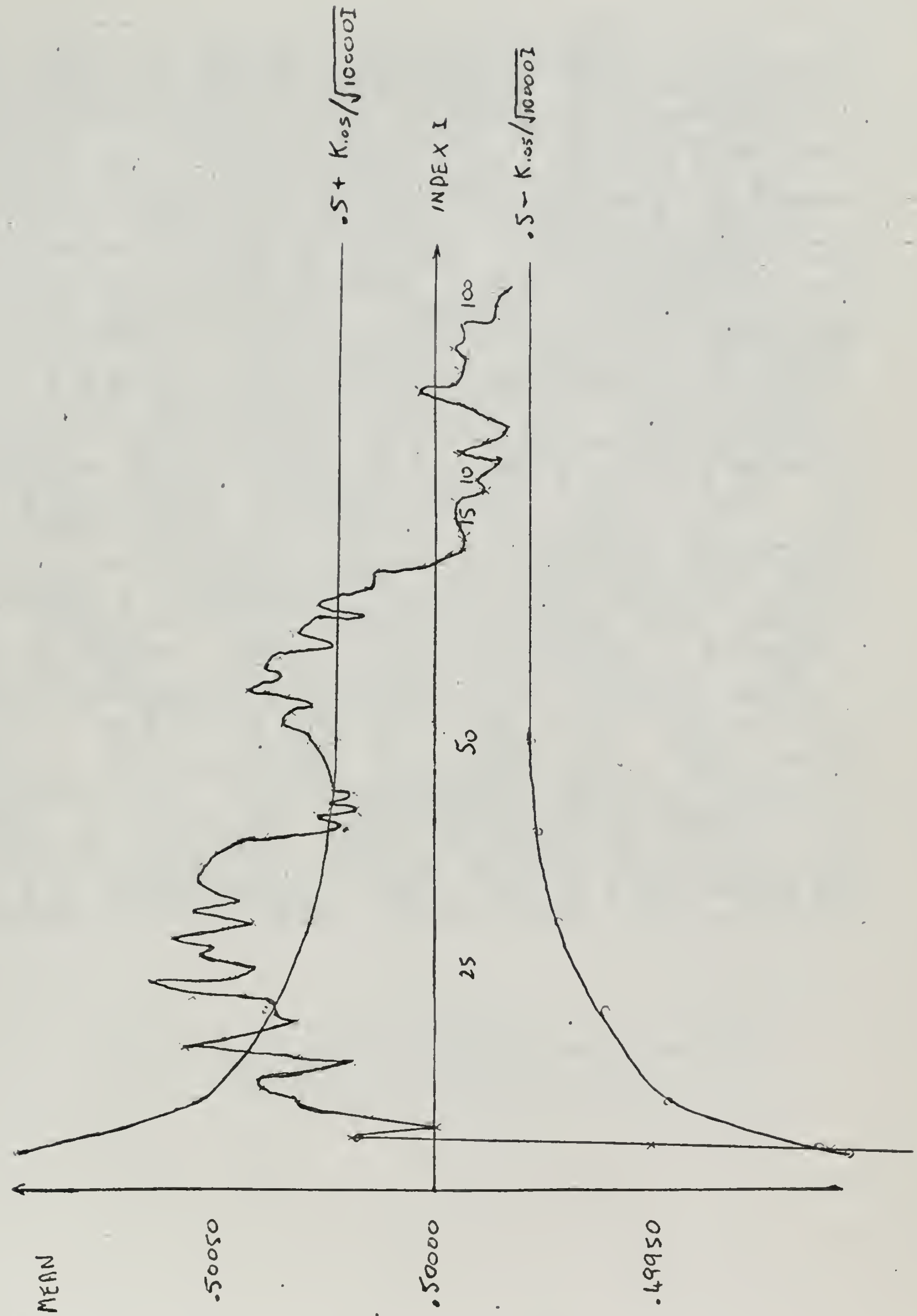
3.2.5 George Marsaglia and M. Donald MacLaren [8] at Boeing Scientific Research Laboratories suggest that the combination of two generators will produce a superior random number of generator. They have tested:

$$A_{R+1} = (2^{17} + 3)A_R \text{ modulus } 2^{35}$$

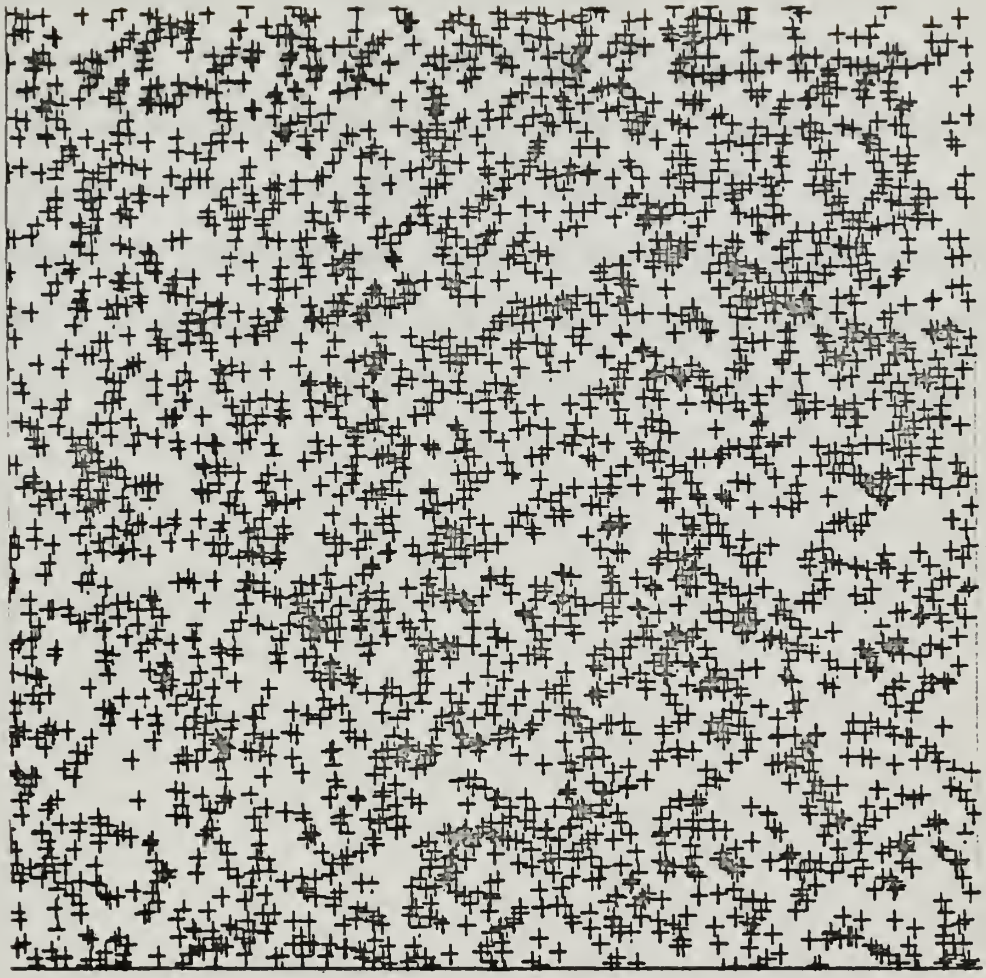
$$\text{and} \quad B_{R+1} = (2^{7} + 1)B_R \text{ modulus } 2^{35}$$

In essence they have used one generator to select numbers from the other. This generator seems to provide an improvement in some local randomness properties. However, the penalty for the improvement is doubling the time of generation. Marsaglia and MacLaren also noted that the method of table look-up may once more become feasible. In the case of the CDC 1604 this method is not practical. However, in a parallel program c omputer a method using a pair of generators may well be advantageous. The generators continually fill up the bottom of a short table in memory as the main program uses numbers from the top. The size of the table is chosen to ensure that the program never uses all the numbers in the table and thus the effective generation time will be just the load cycle time.

3.2.6 The CODAP, Fortran 63, machine language program for the



Cumulative Mean Of the First 'i' Samples
Of The Uniform Random Number Generator
(with 1000 numbers per sample)



Scatter Diagram
For The Uniform (0,1) Random Number Generator
(Scale: 0.2 units per inch)

uniform generator used as a basis for this thesis is in Appendix I. The expected time of generation per number, as calculated over several samples of varying size was found to be 552 microseconds per number. This amounts to producing 1811 numbers per second. However, the generator is theoretically much faster than that. The time per number as calculated from times in the Control Data Corporation specifications for the computer is 121 microseconds. Measured times, depending on the context and the timing mechanism varied from a minimum of 370 to a maximum of 700 microseconds.

4. The Normal Disbribution (Univariate Case).

4.1 Distribution characteristics.

It is desired to generate numbers such that the density function will be

$$f(y) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\left(\frac{y-u}{\sigma}\right)^2\right)$$

where u is the mean of the distribution and σ^2 is the variance. For the basic case the mean is taken to be zero and the variance to be one. The first four moments are M1 = 0, M2 = 1, M3 = 0, M4 = 3. If the desired distribution is to have a mean other than zero, say u, and a variance other than one, say V, then the following transformation applied to the numbers generated by the N(0,1) generator developed here, represented by RN01, will produce a number, RNUV, with the desired characteristics.

RNUV = (RN01) (V) + U

In this paper the normal distribution is treated in three separate sections. The univariate case is developed first, then the bivariate, and finally a general multivariate case is demonstrated. The main purpose of this separate treatment is to allow a more efficient handling of the more commonly used univariate and bivariate cases. A general n-dimensional normal random number generator would be much slower, when used for n equals one, than the univariate generator demonstrated in Section 4. The test procedures for each generator are also different.

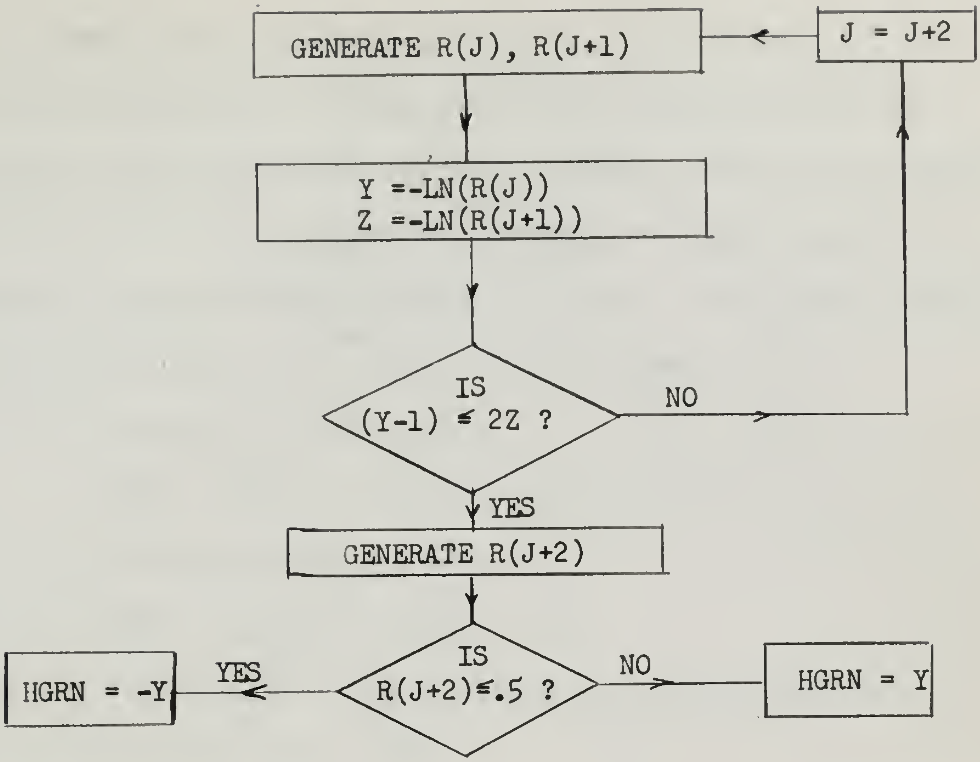
4.2 Methods of generation.

4.2.1 The normal distribution is one of the most used and

tabulated distributions. As with the uniform distribution several procedures have been used to produce random numbers from it. The methods discussed here are those that are most adaptable to use on a computer.

4.2.2 The most common procedure has been to take the sum of K uniform random numbers. The central limit theorem shows that this (with the mean subtracted, and divided by the standard deviation) approaches the normal as K gets large. Vaa tested a generator using the sum of twelve uniform random numbers. This approximation has the disadvantage of being truncated at plus and minus six. Even more important a factor is the time required to generate these numbers. It is hoped that a more exact and faster method can be found.

4.2.2 The so-called half-Gaussian method [11] provides a theoretically exact transformation from the uniform to the normal. However in an attempt to reduce time some fairly drastic approximations have been made. These approximations should not affect randomness, however, and should only be a factor in accuracy beyond the fifth decimal place. The following flow chart is the basis for the routine. R(J) is a uniform random number. HGRN is the normal number.



The routine first generates the positive half of a normal distribution then adds a random sign selected by another uniform number. This program makes no external calls to the log function but rather uses the following series approximation:

$$T = (R-1)/(R+1)$$
$$LN(R)= 2(T+T^3/3+ T^5/5+ T^7/7+ T^9/9)$$

The CODAP function sub-program is Appendix II.

4.2.3 An excellent approximation technique has been developed by Marsaglia and Bray [9] . Marsaglia has developed several similar techniques but the one proposed here seems optimum in terms of time required per number and the storage space required. The method involves selecting one of four functions of varying complexities to produce the random number. One function is very simple and fast and

is used 86% of the time; the next function is also fast and is used 11% of the time. The remaining three percent of the time much more complex functions are used, however, due to the rapidity with which 97% of the numbers are formed, the overall expected generation time per number is relatively low. The program outline is as follows (MSRN is the desired random number):

1. 86.38% of the time, set

   $MSRN = 2[R(J) + R(J+1) + R(J+2) - 1.5]$

2. 11.07% of the time, set

   $MSRN = 1.5[R(J) + R(J+1) - 1]$

3. 2.28002039% of the time form pairs (X,Y) such that

   $X = 6R(J) - 3$ and

   $Y = 0.358\ R(J+1)$

   until $Y \leq G3(X)$; then set MSRN = X. G3(X) is defined by:

$17.49731196\exp(-x^2/2)-4.73570326(3-x^2)-2.15787544(1.5-|x|)$ for $|x|<1$

$17.49731196\exp(-x^2/2)-2.36785163(3-|x|)^2 -2.15787544(1.5-|x|)$ for $1<x<1.5$

$17.49731196\exp(-x^2/2)-2.36785163(3-|x|)$ for $1.5 < x < 3$

4. 0.26997961% of the time form pairs (X,Y) until either X or Y is greater than three, then let that one equal MSRN. For RM(J) uniform on the interval (-1,1) and such that if $RM(J)^2 + RM(J+1)^2 \leq 1$, then let $Z = RM(J)^2 + RM(J+1)^2$

and $$X = RM(J)[\{9-2LN(Z)\}/(Z)]$$

$$Y = RM(J+1)[\{9-2LN(Z)\}/(Z)]$$

The CODAP function sub-program is Appendix III.

4.3 Selection of generators.

A complete table of results of the tests on the normal generators is in Appendix IV. Partial results are presented below.

	HALF-GASSIAN TECHNIQUE			MARSAGLIA TECHNIQUE		
Average observed generation time per number	3625 microseconds			1503 microseconds		
Sample range	1-1000	1-10000	10001-20000	1-1000	1-10000	10001-20000
Sample size	1000	10000	10000	1000∅	10000	10000
Mean (theor. = 0)	0.04	-0.01	0.00	0.00	-0.02	-0.01
Variance (theor. = 1)	1.06	1.04	1.01	.99	1.00	.99
3rd Mom. (theor. = 0)	-0.05	-0.02	0.01	.01	-0.04	-0.07
4th Mom. (theor. = 3)	3.17	3.20	3.12	2.77	2.97	2.93
$\sqrt{N}\ \bar{X}$ ($K_{.05}$=1.64)	1.32	-0.96	0.30	0.13	-2.47	-1.06
$\frac{S-(N-1)}{\sqrt{2(N-1)}}$	1.30	2.57	0.95	-0.14	-0.11	-0.80
D_{max}/N	0.0270	0.0073	0.0033	0.0082	0.0095	0.0049
$1.22/\sqrt{N}$	0.0386	0.0122	0.0122	0.0386	0.0122	0.0122

The samples generated by the half-Gaussian technique passed the hypothesis on the mean, at the 10% level, 80% of the time; on the variance at the 10% level, 30% of the time. At the 5% level the test on the mean was passed 90% of the time, the test on the variance was passed

[illegible]

[illegible]

[illegible]

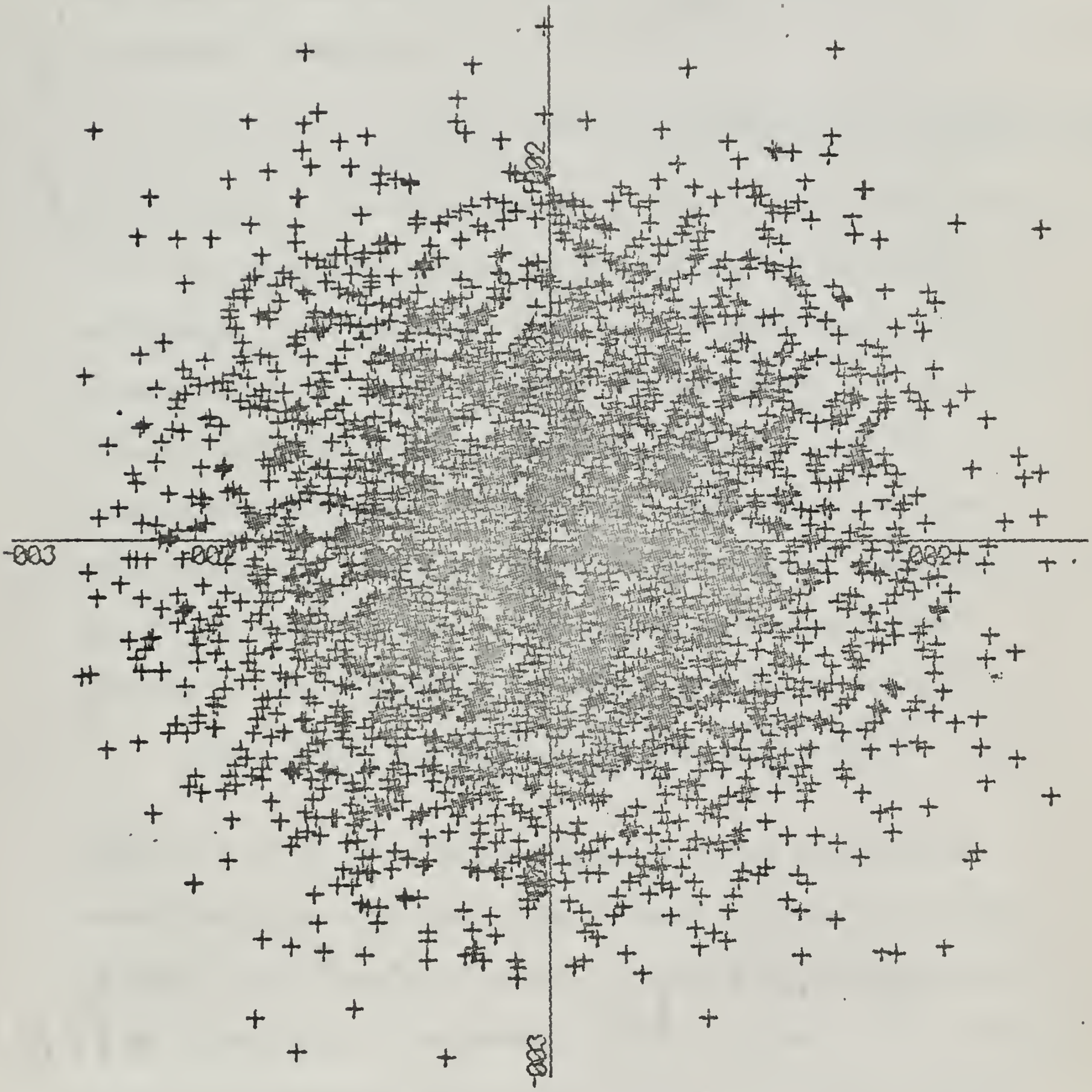
[illegible]

[illegible]

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

[illegible]

50% of the time. The Marsaglia technique generated samples that, at the 10% level, passed the test on the mean 80% of the time, and on the variance 90% of the time. The Marsaglia technique consistently passed the Kolmogorov-Smirnov test but appeared a little heavy in the 'tails' none the less. For a further examination of this see Addendum 1. The half-Gaussian technique failed the Kolmogorov-Smirnov test once but appeared better behaved in the tails. The graph on page 21 is a scatter diagram for the numbers produced by the Marsaglia technique. The graph consists of 4500 points. Marsaglia's technique produced better results for the first four moments. The decisive factor that leads to the selection of one of the generators is the time to generate each number. The Marsaglia technique is more than twice as fast as the half-Gaussian technique, is the one used in further developments, and is the one recommended for general use.



Scatter Diagram
For The Normal Random Number Generator
(Scale: 1.0 units per inch)

5. The Normal Disbribution (Bivariate Case)

5.1 Disbribution characteristics.

It is desired to generate pairs of numbers (X_{1i}, X_{2i}) such that the two-dimensional random variable (X_1, X_2) has the joint density function

$$f(x_1,x_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp\left[-\frac{1}{2(1-\rho^2)}\left(\left(\frac{x_1-u_1}{\sigma_1}\right)^2 - 2\rho\frac{x_1-u_1}{\sigma_1}\frac{x_2-u_2}{\sigma_2} + \left(\frac{x_2-u_2}{\sigma_2}\right)^2\right)\right]$$

for all (x_1,x_2). The constants are u_1 and u_2, the means; σ_1 (>0), σ_2 (>0), the standard deviations; and ρ $(-1 \leq \rho \leq 1)$, the correlation coefficient. Thus the requirement for a random vector must be accompanied by the specification of the mean vector (u_1,u_2), the variances (the squares of the standard deviation), and the correlation coefficient. Another equivalent form of the input would be the mean vector and the covariance matrix. This last form of input will be used in the general multivariate case. The distribution is specified in matrix notation as follows:

$$f(X) = f(x_1,x_2) = \frac{|R|^{1/2}}{(2\pi)^{p/2}} \exp\left\{-\frac{1}{2}(Y-U)'R(Y-U)\right\}$$

where f(X) is the joint density function of the x's; p is the dimension-in this case two; U is the mean vector; and R is the inverse of the convariance matrix. Thus $|R|^{1/2}$ is the square root of the determinant of the inverse of the covariance matrix; (Y-U) 'R(Y-U) is a quadratic form, where (Y-U)' is a one by p vector, R is a p by p matrix, and (Y-U) is a p by one vector.

5.2 Method of generation.

If R(J) is a normal random number, the random bivariate vector (VN1, VN2) is formed as follows:

$$VN1 = (\sigma_1)R(J)+u_1$$

$$VN2 = (\sigma_1)R(J)+(\sigma_2)R(J+1)(\sqrt{1-\rho^2})+U_2$$

The source of the normal random numbers is the Marsaglia routine described in the previous section. The generator is Appendix V.

5.3 Testing the generator.

First the maximum likelihood estimators of the mean vector and the covariance matrix are formed [12] . The maximum likelihood estimators for the parameters are:

$$\hat{u} = (\bar{x}_1 \quad \bar{x}_2)' = (\tfrac{1}{N}\Sigma x_{1i} \quad \tfrac{1}{N}\Sigma_{2i})'$$

$$\hat{\sigma}_1^2 = \frac{1}{N}\left(\Sigma x_{1i} - \bar{x}_1\right)^2 = \frac{1}{N}\left(\Sigma x_{1i}^2 - N\bar{x}_1^2\right)$$

$$\sigma_2^2 = \frac{1}{N}\left(\Sigma x_{2i} - \bar{x}_2\right)^2 = \frac{1}{N}\left(\Sigma x_{2i}^2 - N\bar{x}_2^2\right)$$

$$\rho_{12} = \frac{\Sigma(x_{1i}-\bar{x}_1)(x_{2i}-\bar{x}_2)}{\sqrt{\Sigma(x_{1i}-\bar{x}_1)^2\Sigma(x_{2i}-\bar{x}_2)^2}}$$

$$\hat{\sigma}_{12} = \sqrt{\hat{\sigma}_1^2}\sqrt{\hat{\sigma}_2^2}\,\hat{\rho}_{12} .$$

The distribution of the mean when the covariance matrix is unknown was shown by Hotelling to be a multivariate analogue of the t-test and is called the generalized T statistic. However, the covariance matrix is known and once again we can use a more powerful test.

$H_o : u = (U_{10}\ U_{20})'$ $\qquad$ $H_A : u \neq (U_{10}\ U_{20})'$

Construct the statistic H such that:

$$H = N(x_1-u_{10},\ x_2-u_{20})C^{-1}(x_1-u_{10},\ x_2-u_{20})'$$

where C is the given covariance matrix. If $H < \chi_2^2(\alpha)$ we accept the null hypothesis. The test for a hypothesized mean vector $U = (0\ ,\ 0)'$, and a covariance matrix with variances one and correlation coefficient ρ, reduces to calculating H such that:

$$H = \frac{N}{1-\rho^2}\begin{pmatrix}\bar{x}_1 & \bar{x}_2\end{pmatrix}\begin{pmatrix}\frac{1}{\sigma_1^2} & -\frac{\rho}{\sigma_1\sigma_2} \\ -\frac{\rho}{\sigma_1\sigma_2} & \frac{1}{\sigma_2^2}\end{pmatrix}\begin{pmatrix}\bar{x}_1 \\ \bar{x}_2\end{pmatrix}$$

$$= \frac{N}{1-\rho^2}\left(\bar{x}_1^2 - 2\rho\bar{x}_1\bar{x}_2 + \bar{x}_2^2\right), \quad \text{for } \sigma_1 = \sigma_2 = 1.$$

for sample sizes of 1000 the test will be made for ρ = 0.25, 0.5, 0.75. At the 10% level if $H \leq 4.61$ accept the hypothesis. In order to test whether the covariance matrix is a given matrix, the test as outlined in Anderson [12] could be used, but before this a digression into the philosophy of testing is in order. The purpose of these hypothesis tests is in general to render a judgement as to whether a sample of data from some type of experiment is distributed according to a certain distribution with certain parameters. This type of test was applicable to our uniform random number generator, but its use seems extraneous for the distributions derived from this generator. The transformations from the uniform to the normal to bivariate normal are either theoretically exact or controllably close to being exact. What is really needed is a method to ensure that any 'inaccuracies' there may have been in the basic generator are not somehow amplified in the transformation so as to bias the derived distribution.

[illegible]

$$[illegible]$$

$$[illegible]$$

[illegible]

[illegible]

To this end sophisticated testing procedures do not seem to be in order. Instead the testing will be restricted to calculation of estimators, and some goodness of fit tests. The testing plan for the bivariate generator is to test 10 sets of 1000 each for each of several correlation coefficients. The generator was also tested for some odd combinations of parameters- such as U=(100.0 -100.0), σ_1^2=0.5, σ_2^2=25.0, ρ = -0.8.

With 0.75 as the correlation coefficient, the fourth set showed a disagreeably large difference in the maximum likelihood estimators. The H statistic was close to the critical value at the 10% level. The sample passed the test in 90% of the cases. With 0.50 as the correlation coefficient the sample passed the H test 90% of the time. Sample set four once again had rather low covariance estimators and once again had 3.76 as the value for H (compared with 4.61 for the critical value). Sample set six was again the culprit in not passing the H test. A similar result was noted in the set using 0.25 as a correlation coefficient. This suggests the advisability of further testing of the Marsaglia generator in these ranges. Complete test results are in Appendix VI. The generator performs as expected and is recommended for use.

6. The Normal Disbribution (Multivariate Case).

6.1 Distribution characteristics.

As is noted in Section 6.1 the desired distribution is

$$f(X) = f(x_1, x_2, \ldots, x_N) = \frac{|R|^{1/2}}{(2\pi)^{N/2}} \exp\left(-\tfrac{1}{2}(Y-U)'R\,(Y-U)\right)$$

where R is the inverse of the covariance matrix.

6.2 Method of generation.

As shown by Wold [16] , the method is based on a triangularization of the covariance matrix such that if:

$$\begin{bmatrix} C11 & C12 & \cdots & C1N \\ C21 & C22 & \cdots & C2N \\ \vdots & & \ddots & \vdots \\ CN1 & CN2 & \cdots & CNN \end{bmatrix} = \begin{bmatrix} P11 & 0 & \cdots & 0 \\ P21 & P22 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ PN1 & PN2 & & PNN \end{bmatrix} \begin{bmatrix} P11 & P21 & \cdots & PN1 \\ 0 & P22 & \cdots & PN2 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & PNN \end{bmatrix}$$

then the N-dimensional random vector Z may be formed as follows. The RN(I) are the normal random numbers generated by the Marsaglia technique.

Z(1) = P11x RN(1)

Z(2) = P21 x RN(1) + P22 x RN(2)

Z(3) = P31 x RN(1) + P32 x RN(2) + P33 x RN(3)

⋮

Z(N) = PN1 x RN(1) + PN2 x RN(2) + ... + PNN x RN(N).

The method as programmed, assumes all the means are zero, but simple addition of the mean when required will remedy this. The matrix triangularization is based on a symmetric, positive definite or positive semi-definite matrix C. Thus any pair of the random variables can have a partial correlation coefficient of one. The routine will set all elements of the vector that are dependent equal to zero. This procedure was selected since in order to relate the

variables properly requires an inordinately extra amount of computer time- time that even when not needed adds to execution time. If the user desires some variable to be a linear transformation of some others, then he only needs to keep track of which variables these are, and where the program sets the vector element equal to zero, substitute the appropriate linear combination of the corresponding independent elements of the vector. The routine also checks and where the dependence is very close to one will assume a correlation of one, and proceed as noted above. This procedure is required to prevent division by numbers very close to zero. The following example will clarify the above explanation. Suppose it is desired to generate random vectors from a distribution with mean vector one and covariance matrix C, where:

$$C = \begin{bmatrix} 2 & 1 & 3 \\ 1 & 4 & 5 \\ 3 & 5 & 8 \end{bmatrix}$$

Thus column three is a linear combination, in fact the sum, of columns one and two. This means that the third variable is not an independent variable. The triangulation routine will produce a matrix P of the form:

$$P = \begin{bmatrix} a & o & o \\ b & d & o \\ c & e & o \end{bmatrix}$$

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

As expected column three is identically zero. Thus if the first two normal (0,1) numbers generated are denoted by R1 and R2, the generated vector will be of the form:

$$Z = (aR1, \quad bR1 + dR2, \quad 0) \quad .$$

Since it has been determined that the third variable is the sum of the first two and also that we desire all the means to be one then the desired vector is of the form:

$$Z = (aR1 + 1, \quad bR1 + dR2 + 1, \quad aR1 + bR1 + dR2 + 1)$$

The generator is Appendix VII.

6.3 Testing the generator.

The maximum likelihood estimators of the mean vector and the covariance matrix are formed as follows:

$$BM1(J) = \frac{1}{M} \sum_{I=1}^{M} Y(I,J) \qquad \text{For } J = 1, 2, \ldots, N$$

$$C(I,J) = \frac{1}{M} \left[\sum_{K=1}^{M} \left\{ Y(I,K) \cdot Y(J,K) \right\} \right] \qquad \text{for } I, J = 1, 2, \ldots, N.$$

N is the dimension of the covariance matrix, M is the number of sample vectors, the Y(I,J) are the vector elements, the BM1(J) are the elements of the mean vector estimates, and the C(I,J) are the elements of the covariance matrix estimate. The results of the tests for various covariance matrices are listed in Appendix VIII.

7. The Poisson Distribution.

7.1 Distribution characteristics.

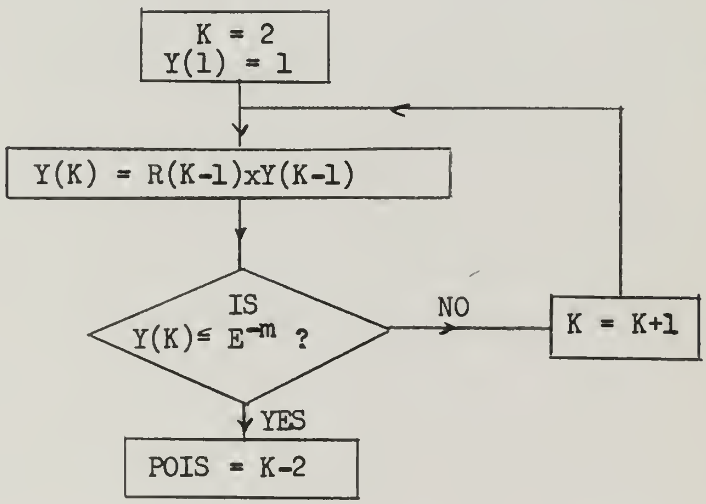
It is desired to generate numbers such that:

$$P\left[x = a\right] = \frac{e^{-m} m^{a}}{a!} \quad \text{for } m > o \; ; \; a = a\ 1, \ldots$$

Thus the Poisson distribution is a discrete distribution for integer values of a, and is characterized by the parameter m. The first four moments are M1 = M2 = M3 = m, and M4 = $3m^2$+m.

7.2 Method of generation.

This method is due to Kahn [11] and is a theoretically exact transformation. The flow chart of the routine is shown below:



POIS is the generated Poisson random number, m is the parameter, E is the irrational number 2.71828....

7.3 Testing the generator.

The first four moments are calculated for samples of 5000 or 1000 with the parameter m taking on various values. The Kolmogorov-Smirnov goodness of fit test is applied to some of the

samples. As Tate and Clelland [19] have stated the test is applicable to discrete distributions with neglible changes in significance for large sample sizes. The generator performed well and consistently passed the test. The test results are in Appendix X, while the generator itself is in Appendix IX.

8. The Exponential Disbribution.

8.1 Distribution characteristics.

It is desired to generate random numbers such that the density function is: $f(a) = e^{-a}$.

The first four moments are M1 = M2 = 1, M3 = 2, M4 = 9. The distribution is often specified as:

$$f(a) = \lambda e^{-\lambda a} \quad \text{for } \lambda > 0, a \geq 0$$

where λ is the parameter of the distribution. The generator here takes the case where $\lambda = 1$. However the exponential distribution has the characteristic that if the numbers generated here, (Expl) for which $\lambda = 1$, are simply multiplied by the parameter desired for the distribution (LAMBDA), then the desired numbers are generated.

EXPL = EXP1 * LAMBDA

8.2 Method of generation.

This method is a theoretically exact procedure due to Marsaglia [13] . A more obvious method would be to take the integral transformation- the negative logarithm of a uniform random number. However this method is slower on most computers than the one demonstrated here. Let C = 1/(e-1), and let the random variable N take on the values 1,2,3,4,... with probabilities c, c/2!,c/3!,... Then let the random variable M take values 0,1, 2,3,... with probabilities 1/ce, $1/ce^2$, $1/ce^3$,... Then we form the desired random number:

EXP1 = M+MIN(U1, U2,...,UN)

The CODAP generating routine is Appendix XI. The sample moments and the results of the goodness of fit tests are in Appendix XII.

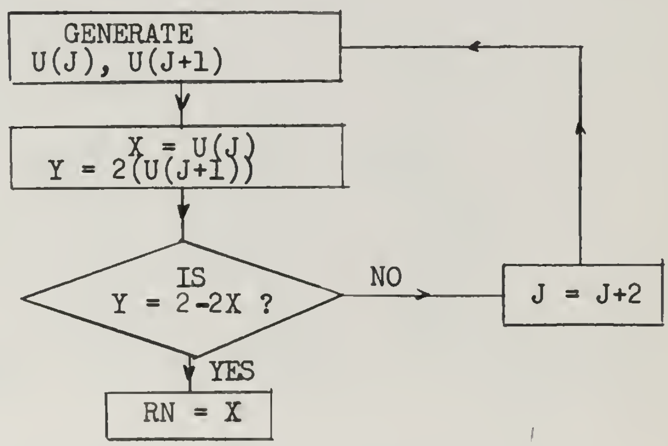
9. A General Disbribution.

The method of obtaining random numbers described in this section is applicable only to a restricted set of distributions. However, the method is applicable to a type of distribution that frequently occurs in model building and war gaming. The method is examined by use of an example. A discussion of how and when to use this method for other distributions is included.

It is desired to produce numbers from the density function f(x) such that:

$$f(x) = 2-2x \quad \text{for } 0 \leq x \leq 1$$

The method is based on drawing uniform numbers in pairs, normalizing the scale of the numbers, and testing to see if the point formed by the pair lies under the curve described by the density function. If it does, the x coordinate of the point is taken as the random number; if it does not, another pair of uniform numbers are drawn and the process is continued. The flow chart for f(x) = 2-2x is:



The U(J) are the uniform (0,1) random numbers and RN is the random

number from the distribution f(X). Thus the method could be applied with theoretical exactness to any bounded continuous distribution. Another commonly used density is g(x) where

$$g(x) = nx^{n-1} \qquad \text{for } 0 < x < 1, \quad n \geq 1$$

This density function is bounded and continuous and a member of the class to which this method is applicable. In many applications the user must make a judgement about the amount of use a generator is going to get. If it is intended for heavy use it may be well to explore the literature for, or to design, a more efficient generator than the type described in this section. However, if the generator is to only be used a limited amount, or if only a restricted amount of resources are available, this generator is easily programmed and will be eminently satisfactory in a wide variety of cases. The efficiency of the generator may be defined as the reciprocal of the expected number of iterations needed to produce a point under the curve.

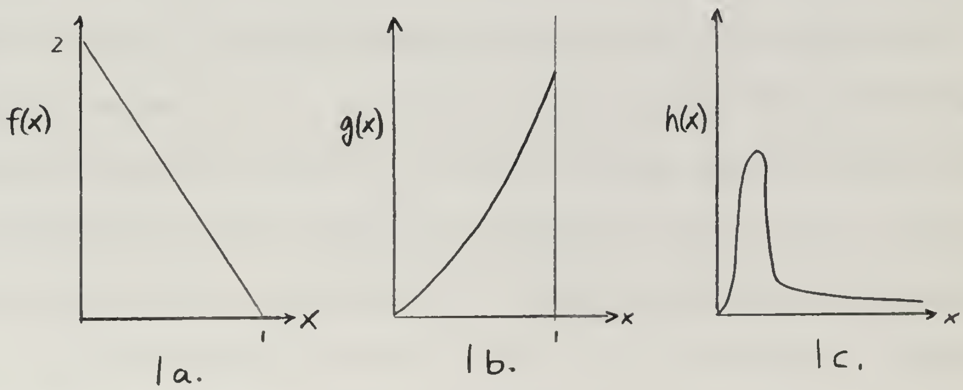
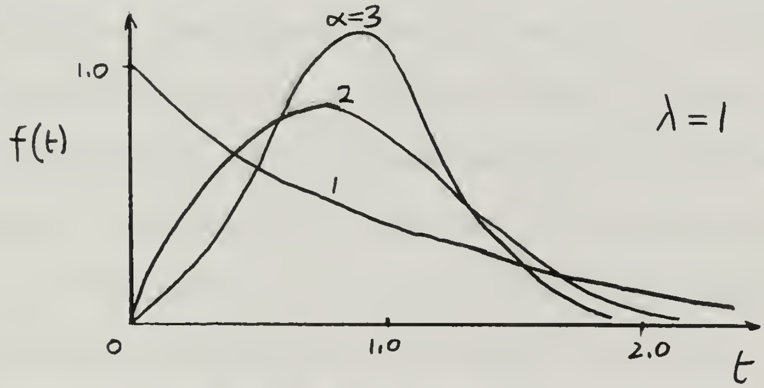


Figure 1a clearly has an efficiency of 1/2, figure 1b has an efficiency of less than 1/2, and as n gets large the efficiency decreases rapidly.

A distribution such as figure 1c where h(x) goes to zero at some large value of x may vary well have such a low efficiency as to make this generator unsuitable. If h(x) only approaches zero as x becomes large, and therefore x is not bounded, then another generator should be used. However, if one cannot be found and the user is not too concerned with the tail of the distribution, we can easily adapt this generator. Suppose we wished to generate numbers for the Weibull distribution, which is a three parameter distribution used widely in reliability theory, then the following procedure might be followed:



Since the distribution may take any of the forms in figure 2, we must first limit the development to those parameter combinations that are bounded at x = 0. Since the user is also often concerned with the behaviour in part of the tail it must first be decided if the distribution could simply be truncated at some point. Even if this is acceptable an efficiency of .00001 can easily be envisioned. If it is unacceptable, the tail beyond some point could be closely approximated by the exponential distribution.

10. Summary and Conclusions.

The analyst who desires to use any of the generators demonstrated here is once again warned that although the generators are recommended for general use they all have aberrations of some kind. The general form of these aberrations is noted where known. If a user suspects from analysis of his results that the aberration in the generator is influencing his results then it is recommended that he first modify the uniform generator by one of the methods suggested in section two. The chance of this occuring is remote and other parts of the model should be checked carefully.

The programs demonstrated here are very fast. The problem of speed is stressed here and is a major decision criterion in the selection of generators. In some applications the speed may not be as important a factor, and in this case the type of generator discussed in section nine may be very cost effective in that little investment in programming and testing is required. The problem of measuring the speed of generation of a number is not as simple as it may appear. The generation time depends on the program using the generator and also on the method used to time the routine. The time to generate a number based on the Control Data Corporation specifications for the 1604 computer theoretically should be 121 microseconds. The observed generation times vary from 370 to 700 microseconds.

The tests used here are statistically sound, but the meaning of the results of several tests of the same generated sample could bear further investigation.

Some very interesting new methods for generating random numbers from various distributions have been developed by H. Rubin. Rubin's work is soon to appear as a Stanford Applied Statistics Laboratory Technical Report. It would be of interest to compare his methods with the technique used above.

Bibliography

1. Barron, J. M., Random Number Generation on the CDC 1604, U. S. Naval Postgraduate School Thesis, 1962.

2. Vaa, Norman A., Generation and Testing of Random Numbers of an Arbitrary Distribution, U. S. Naval Postgraduate School Thesis, 1962.

3. Hull, T. E. and Dobell, A. R., Mixed congruential random number generator for binary machines, Journal of the Association for Computing Machinery, v. 11, January, 1964: 31-40.

4. Burington, R. S. and May, D. C., Handbood of Probability and Statistics, Handbook Publishers Inc., 1953.

5. Mood, A. M. and Graybill, F. A. Introduction to the Theory of Statistics, Chap. 12, McGraw Hill Book Co., 1963.

6. Massey, Frank J., The Kolmogorov-Smirnov Test for goodness of fit, Journal of the American Statistical Association, v. 48, 1951.

7. Smirnov, N., Table for estimating the goodness of fit of empirical distributions, Annals of Mathematical Statistics, v. 19, 1948: 279-281.

8. MacLaren, M. Donald and Marsaglia, George, Uniform random number generators, Journal for the Association for Computing Machinery, v. 12, January, 1965: 83-89.

9. Marsaglia, George and Bray, T. A., A small procedure for generating normal random variables, Boeing Scientific Research Laboratories Mathematical Note No. 283, November, 1962. (AD 294455).

10. Box, G. P. and Muller, M. E., A note on the generation of random normal deviates, Annals of Mathematical Statistics, v.29, 1958: 610-611.

11. Kahn, H., Applications of Monte-Carlo, RM 1237-AEC, April 1956.

12. Anderson, T. W., An Introduction to Multivariate Statistical Analysis, John Wiley and Sons, 1960.

13. Marsaglia, G., Generating exponential random variables, Annals of Mathematical Statistics, v. 32, 1961: 899-900.

14. Mann, H. B. and Wald A., On the choice of the number of class intervals in the application of the chi-square test, Annals of Mathematical Statistics, v. 13, 1942: 306-317.

15. Williams, C. Arthur, On the choice of the number and width of classes for the chi-squared test of goodness of fit, Journal of the American Statistical Association, v. 45, 1950: 77-86.

16. Wold, H., Tracts for Computers, No. XXV, Cambridge Press, 1948.

17. Greenberger, Martin, Method in Randomness, Communications of the Association for Computing Machinery, v. 8, 1965: 177-179.

18. Lach, J., Random number generators, Yale Computer Ctr. Mem. 27, October, 1963.

19. Tate, M. W. and Clelland, R. C., Non-Parametric and Shortcut Statistics, Interstate Printers and Publishers Inc., 1959.

Appendix I.

The Fortran 63 CODAP function subprogram for generating uniform (0,1) random numbers.

The program is called by using a variable, 'UNIFORM(DUMMY)'. The argument 'DUMMY' is not used. For example: A = UNIFORM(DUMMY)+ B.

No common or dimension entries are required.

The observed average length of the program is 552 microseconds.

The starting number may be change to 'R   OCT   5402033450727422' if it is desired to enter the generator near the middle of the period.

```
               IDENT     UNIFORM
               ENTRY     UNIFORM
M1             OCT       4000000000000000
M2             OCT       2000000000000000
R              OCT       1777777777777777      = (2^48 - )
UNIFORM        SLJ       **
               LDA       *
               ALS       24
               INA       1
               SAU       EXIT
               ENQ       0
               LDA       R
               LLS       +19
               SCL       M1
               ADD       R
               ADD       R
               ADD       R
               STA       R
               ARS       11
               ADD       M2
               FAD       M2
EXIT           SLJ       **

               END
```

Appendix II.

The half-Gaussian technique for producing normal random numbers.

The number is generated by the use of the expression 'GSRN(BLK)'.

The number is returned as GSRN. The argument 'BLK' is not used.

No external calls are made.

The observed average length of time to generate one number is 3625 microseconds.

```
        IDENT   GSRN
        ENTRY   GSRN
GSRN    SLJ     **
        LDA     *
        ALS     24
        INA     1
        SAU     GSRN
.190    RTJ     UNIFORM
        STA     B1
        FAD     =02001400000000000
        STA     TEMP
        LDA     B1
        FSB     =02001400000000000
        FDV     TEMP
        STA     X
        FMU     X
        STA     X2
        FMU     =01774707070705726
        FAD     =01775444444443023
        FMU     X2
        FAD     =01775631463146315
        FMU     X2
        FAD     =01776525252524341
        FMU     X2
        FAD     =02001400000000000
        FMU     X
        FMU     =05775377777777777
        STA     Y
        RTJ     UNIFORM
        STA     B1
        FAD     =02001400000000000
        STA     TEMP
        LDA     B1
        FSB     =02001400000000000
        FDV     TEMP
```

```
          STA     X
          FMU     X
          STA     X2
          FMU     =01774707070705726
          FAD     =01775444444443023
          FMU     X2
          FAD     =01775631463146315
          FMU     X2
          FAD     =01776525252524341
          FMU     X2
          FAD     =02001400000000000
          FMU     X
          FMU     =05774377777777777
          STA     Z
          LDA     Y
          FSB     =02001400000000000
          STA     YM1
          FMU     YM1
          THS     Z
          SLJ     .190
          RTJ     UNIFORM
          THS     =02000400000000000
          SLJ     *+2
          LDA     Y
          SLJ     GSRN
          LAC     Y
UNIFORM   SLJ     GSRN
          SLJ     **
          ENQ     0
          LDA     R
          LLS     19
          SLC     =04000000000000000
          ADD     R
          ADD     R
          ADD     R
          STA     R
          ARS     11
          ADD     =02000000000000000
          FAD     =02000000000000000
          SLJ     UNIFORM
R         OCT     17777777777777777
B1        BSS     1
TEMP      BSS     1
X         BSS     1
Y         BSS     1
YM1       BSS     1
Z         BSS     1
X2        BSS     1
X3        BSS     1
X5        BSS     1
X7        BSS     1
X9        BSS     1
          END
```

Appendix III.

The number is generated by use of the variable 'MARS(ZQ)'. The argument 'ZQ' is not used. For example: 'A = MARS(ZQ)/9.

External calls are made to LOGF, SQRTF, and EXPF.

No common dimension entries are required.

The observed average length of time to generate one number is 1503 microseconds.

```
            IDENT   ZMARS
            ENTRY   ZMARS
UNIFORM     SLJ     **
            ENQ     0
            LDA     R
            LLS     19
            SCL     =04000000000000000
            ADD     R
            ADD     R
            ADD     R
            STA     R
            ARS     11
            ADD     =02000000000000000
            FAD     =02000000000000000
            SLJ     UNIFORM
R           OCT     1777777777777777
ZMARS       SLJ     **
            LDA     *
            ALS     24
            INA     1
            SAU     Z MARS
            RTJ     UNIFORM
G1          THS     P1
            SLJ     G2
            RTJ     UNIFORM
            STA     NORM
            RTJ     UNIFORM
            FAD     NORM
            STA     NORM
            RTJ     UNIFORM
            FAD     NORM
            FSB     =02001600000000000
            STA     NORM
            FAD     NORM
            SLJ     ZMARS
```

```
G2          THS       P2
            SLJ       G3
            RTJ       UNIFORM
            STA       NORM
            RTJ       UNIFORM
            FAD       NORM
            FSB       =02001400000000000
            FMU       =02001600000000000
            SLJ       ZMARS
G3          THS       P3
            SLJ       G4
G31         RTJ       UNIFORM
            FMU       =02004600000000000
            FSB       =02002600000000000
            STA       T
            AJP    2  G3POS
            SCM       =07777777777777777
G3POS       STA       T+1
            FMU       T+1
            STA       T+2
            FMU       =05777377777777777
            CALL      EXPF
            FMU       C1
            STA       T+3
            LDA       T+1
            THS       =02001400000000000
            SLJ       T1.5
            LDA       =02002600000000000
            FSB       T+2
            FMU       C2
            FAD       T+3
            STA       T+3
            LDA       =02001600000000000
            FSB       T+1
            FMU       C3
            FAD       T+3
            SLJ       G3END
T1.5        THS       =02001600000000000
            SLJ       T3.0
            LDA       =02002600000000000
            FSB       T+1
            STA       T+4
            FMU       T+4
            FMU       C4
            FAD       T+3
            STA       T+3
            LDA       =02001600000000000
            FSB       T+1
            FMU       C3
            FAD       T+3
            SLJ       G3END
```

```
T3.0     LDA     =02002600000000000
         FSB     T+1
         STA     T+4
         FMU     T+4
         FMU     C4
         FAD     T+3
G3END    STA     T+4
         RTJ     UNIFORM
         FMU     C5
         THS     T+4
         SLJ     G31
         LDA     T
         SLJ     ZMARS
G4       RTJ     UNIFORM
         FMU     =02002400000000000
         FSB     =02001400000000000
         STA     T
         FMU     T
         STA     T+1
         RTJ     UNIFORM
         FMU     =02002400000000000
         FSB     =02001400000000000
         STA     T+2
         FMU     T+2
         FAD     T+1
         THS     =02001400000000000
         SLJ     G4
         STA     T+1
         CALL    LOGF
         STA     T+3
         FAD     T+3
         SCM     =07777777777777777
         FAD     =02004440000000000
         FDV     T+1
         CALL    SQRTF
         STA     T+3
         FMU     T
         STA     T+4
         AJP  2  *+1
         SCM     =07777777777777777
         THS     =02002600000000000
         SLJ     GOOD
         LDA     T+3
         FMU     T+2
         STA     T+4
         AJP  2  *+1
         SCM     =07777777777777777
         THS     =02002600000000000
         SLJ     GOOD
```

```
          SLJ     G4
GOOD      LDA     T+4
          SLJ     ZMARS
C1        DEC     17.49731196
C2        DEC     -4.73570326
C3        DEC     -2.15787544
C4        DEC     -2.36785163
C5        DEC     0.358
P1        DEC     0.8638
P2        DEC     0.9745
P3        DEC     0.9973002039
T         BSS     5
NORM      OCT     0
          END
```

Appendix IV.

Test results for Marsaglia normal generator.

The test consists of ten consecutive samples with 10000 numbers in each.

MEAN	M2	M3	M4	$\sqrt{N}\bar{X}$	$\frac{S-(N-1)}{\sqrt{2(N-1)}}$	d/n
.0	1.0	.0	3.0	-1.645 to 1.645	-1.645 to 1.645	
-.02474	.99844	-.04435	2.96870	-2.4737	-.1104	0.0095
-.01062	.98869	-.07487	2.92989	-1.0625	-.7996	0.0049
-.00564	.99388	-.07973	2.93538	- .5640	-.4330	0.0027
-.00873	1.00452	-.09225	2.99577	- .8729	.3196	0.0060
-.00921	.99945	-.04774	3.03286	- .9209	-.0392	0.0040
-.00683	.99709	-.10706	3.08487	- .6834	-.2057	0.0048
.00028	.99963	-.04737	3.03274	.0277	-.0264	0.0030
.00732	1.01508	-.04782	3.16472	.7317	1.0664	0.0041
-.00410	1.02454	-.05629	3.09820	- .4104	1.7352	0.0050
-.01732	.97325	-.05067	2.79351	-1.7319	-1.8911	0.0050

Appendix IV. (Cont'd)

Test results for half-Gaussian method.

The test consists of ten consecutive samples with 10000 numbers in each.

MEAN	M2	M3	M4	$\sqrt{N}\bar{X}$	$\frac{S-(N-1)}{\sqrt{2(N-1)}}$	d/n
.0	1.0	.0	3.0	-1.645 to 1.645	-1.645 to 1.645	
-.00961	1.03629	-.01728	3.19774	-.9614	2.5661	0.0073
.00301	1.03418	.00973	3.12078	-.3008	.9529	0.0033
-.01107	1.02600	-.01559	3.12095	-1.1066	1.8386	0.0056
.00844	1.04205	-.02587	3.23017	.8443	2.9735	0.0072
.00230	1.01410	-.00458	3.13768	.2295	.9969	0.0044
-.00475	1.02696	-.00209	3.27642	-.4747	1.9064	0.0036
-.02246	1.04439	.01642	3.29951	-2.2459	3.1384	0.0149
-.00372	1.02968	-.01805	3.13442	-.3716	2.0986	0.0065
-.00684	1.04955	.04399	3.28683	-.6841	3.5036	0.0076
.01761	1.01793	-.00633	3.11080	1.7614	1.2680	0.0084

[illegible]

[illegible]

[illegible]

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Appendix V.

The bivariate normal random number generator (using Marsaglia's technique).

The number is generated by a call 'CALL AKHN(VN1, VN2)'. The bivariate vector is returned as the arguments VN1, VN2.

Subroutine AKHN has the following arguments in common: SIG1SQ, SIG2SQ, RHO, U1, U2. The SIGISQ are the desired variances, RHO is the correlation coefficient, and U1 and U2 are the desired means. Thus besides reading in these values in the main program they must be communicated to the subroutine by a common statement such as: 'COMMON/SIG1SQ/SIG1SQ/SIG2SQ/SIG2SQ/RHO/RHO/U1/U1/U2/U2'.

External calls are made to LOGF, SQRTF, and EXPF.

The observed average generation time for a pair of numbers is 6480 microseconds.

```
            IDENT    AKHN
SIG1SQ      BLOCK    1
            COMMON   SIG1SQ(1)
SIG2SQ      BLOCK    1
            COMMON   SIG2SQ(1)
RHO         BLOCK    1
            COMMON   RHO(1)
U1          BLOCK    1
            COMMON   U1(1)
U2          BLOCK    1
            COMMON   U2(1)
            ENTRY    AKHN
AKHN        SLJ      **
            LDA      *
            ALS      24
            SAU      *+2
+           INA      1
            SAU      EXIT
+           LDA      **
            SAL      VN2
```

```
         ALS     24
         SAL     VN1
+        RTJ     ZMARS
+        STA     A1
         LDA     SIG1SQ
+        CALL    SQRTF
+        FMU     A1
         STA     A2
VN1      FAD     U1
         STA     VN1
         LDA     A2
         FMU     RHO
         FAD     U2
         STA     A2
         LAC     RHO
         FMU     RHO
         FAD     =02001400000000000
         FMU     SIG2SQ
+        CALL    SQRTF
+        STA     A1
+        RTJ     ZMARS
+        FMU     A1
VN2      FAD     A2
         STA     VN2
EXIT     SLJ     **
ZMARS    SLJ     **
         RTJ     UNIFORM
G1       THS     P1
         SLJ     G2
         RTJ     UNIFORM
+        STA     NORM
         RTJ     UNIFORM
+        FAD     NORM
         STA     NORM
         RTJ     UNIFORM
+        FAD     NORM
         FSB     =02001600000000000
         STA     NORM
         FAD     NORM
         SLJ     ZMARS
G2       THS     P2
         SLJ     G3
         RTJ     UNIFORM
+        STA     NORM
         RTJ     UNIFORM
+        FAD     NORM
         FSB     =02001400000000000
         FMU     =02001600000000000
```

```
         SLJ        ZMARS
G3       THS        P3
         SLJ        G4
G31      RTJ        UNIFORM
+        FMU        =0200460000000000
         FSB        =0200260000000000
         STA        T
         AJP   2    G3POS
         SCM        =0777777777777777
G3POS    STA        T+1
         FMU        T+1
         STA        T+2
         FMU        =0577737777777777
         CALL       EXPF
+        FMU        C1
         STA        T+3
         LDA        T+1
+        THS        =0200140000000000
         SLJ        T1.5
+        LDA        =0200260000000000
         FSB        T+2
         FMU        C2
         FAD        T+3
         STA        T+3
         LDA        =0200160000000000
         FSB        T+1
         FMU        C3
         FAD        T+3
         SLJ        G3END
T1.5     THS        =0200160000000000
         SLJ        T3.0
+        LDA        =0200260000000000
         FSB        T+1
         STA        T+4
         FMU        T+4
         FMU        C4
         FAD        T+3
         STA        T+3
         LDA        =0200160000000000
         FSB        T+1
         FMU        C3
         FAD        T+3
         SLJ        G3END
T3.0     LDA        =0200260000000000
         FSB        T+1
         STA        T+4
         FMU        T+4
         FMU        C4
         FAD        T+3
```

```
G3END    STA      T+4
         RTJ      UNIFORM
+        FMU      C5
+        THS      T+4
         SLJ      G31
+        LDA      T
         SLJ      ZMARS
G4       RTJ      UNIFORM
+        FMU      =02002400000000000
         FSB      =02001400000000000
         STA      T
         FMU      T
         STA      T+1
         RTJ      UNIFORM
+        FMU      =02002400000000000
         FSB      =02001400000000000
         STA      T+2
         FMU      T+2
         FAD      T+1
+        THS      =02001400000000000
         SLJ      G4
         STA      T+1
         CALL     LOGF
+        STA      T+3
         FAD      T+3
         SCM      =07777777777777777
         FAD      =02004440000000000
         FDV      T+1
         CALL     SQRTF
+        STA      T+3
         FMU      T
         STA      T+4
         AJP    2 *+1
         SCM      =07777777777777777
+        THS      =02002600000000000
         SLJ      GOOD
+        LDA      T+3
         FMU      T+2
         STA      T+4
         AJP    2 *+1
         SCM      =07777777777777777
+        THS      =02002600000000000
         SLJ      GOOD
+        SLJ      G4
GOOD     LDA      T+4
         SLJ      ZMARS
UNIFORM  SLJ      **
         ENQ      0
         LDA      R
```

```
          LLS     19
          SCL     =0400000000000000
          ADD     R
          ADD     R
          ADD     R
          STA     R
          ARS     11
          ADD     =0200000000000000
          FAD     =0200000000000000
          SLJ     UNIFORM
C1        DEC     17.49731196
C2        DEC     -4.73570326
C3        DEC     -2.15787544
C4        DEC     -2.36785163
C5        DEC     0.358
P1        DEC     0.8638
P2        DEC     0.9745
P3        DEC     0.9973002039
T         BSS     5
NORM      OCT     0
R         OCT     1777777777777777
A1        BSS     1
A2        BSS     1
          END
```

Appendix VI.

Test results for bivariate normal generator.

The test consists of ten consecutive samples of 1000 numbers each for each of three different correlation coefficients.

THEORETICAL					MAXIMUM LIKELIHOOD ESTIMATES					
COVARIANCES		CORR. COEFF.	MEANS		COVARIANCES		CORR. COEFF.	MEANS		H
1.0	1.0	1.0	1.0	1.0	.9820	.9820	1.0000	.9672	.9672	
1.0	1.0	0.75	0.0	0.0	.9311	.9914	.7313	-.0093	.0047	.20
					1.0318	1.0417	.7731	-.0562	-.0464	1.56
					.9039	.8701	.7347	-.0407	-.0565	1.60
					.8551	.8862	.7340	-.0689	-.0866	3.76
					1.0332	1.0215	.7486	-.0252	-.0313	.49
					1.0300	1.0098	.7535	-.0861	-.1265	8.09
					.9627	1.0034	.7350	-.0538	-.0153	.17
					.9655	1.1056	.7606	-.0003	-.0021	.00
					1.0732	1.0957	.7639	-.0523	-.0348	1.39
					1.0387	1.0587	.7550	.0430	.0366	.95
1.0	1.0	0.50	0.0	0.0	.9311	1.0290	.4813	-.0093	.0107	.20
					1.0318	1.0138	.5398	-.0562	-.0337	1.60
					.9039	.8793	.4682	-.0407	-.0543	1.60
					.8651	.9090	.4787	-.0689	-.0802	3.76
					1.0332	1.0206	.4956	-.0252	-.0288	.49
					1.0300	1.0005	.5021	-.0861	-.1241	8.09
					.9627	1.0320	.4834	-.0538	.0059	2.17
					.9655	1.1277	.5401	-.0003	-.0026	.00
					1.0732	1.0822	.5265	-.0523	-.0204	1.39
					1.0387	1.0570	.5115	.0430	.0272	.95

THEORETICAL COVARIANCES		CORR. COEFF.	MEANS		MAXIMUM LIKELIHOOD ESTIMATES COVARIANCES		CORR. COEFF.	MEANS		H
1.0	1.0	0.25	0.0	0.0	.9311	1.0497	.2417	-.0093	.0148	.20
					1.0318	.9844	.2961	-.0562	-.0030	1.60
					.9039	.8970	.2080	-.0407	-.0482	1.60
					.8652	.9255	.2316	-.0689	-.0683	3.76
					1.0332	1.0227	.2429	-.0252	-.0244	.49
					1.0300	.9967	.2483	-.0861	-.1122	8.09
					.9627	1.0490	.2400	-.0538	-.0232	2.17
					.9655	1.1124	.3202	-.0003	-.0028	.00
					1.0732	1.0631	.2834	-.0523	-.0067	1.39
					1.0387	1.0501	.2664	.0430	.0172	.95

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]		
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]		
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]		
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]		
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]		
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]		
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]		
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]		
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]		

Appendix VII

The FORTRAN programs to produce multivariate normal random vectors. The first entry must be 'CALL TRIANG'. This produces the triangularized matrix, and allows repetitive calls to 'CALL MULTN(Z)'. The argument 'Z' is the starting address of the random vector. Several other entries are necessary. The dimension of the covariance matrix (and the desired vectors) is set equal to 'NR'. The desired covariance matrix is stored as a matrix called 'C'. A common statement with 'NR' and 'C', with 'C' appropriately dimensioned, is included. 'Z' is also dimensioned. The following sample program will read in a matrix 'C', which is a 5 by 5 matrix punched column by column on cards. The random vectors are stored in a 5 by 100 array. Subroutine 'MULTN' and subroutine 'TRIANG' follow. Function 'ZMARS', the normal random number generator, from Appendix III must be added. Subroutine TRIANG makes external calls to SQRTF.

```
PROGRAM EXAMPLE
     COMMON/NR/NR/C/C(5,5)/P/P(5,5)
     DIMENSION Z(5), ARRAY(100,5)
     NR=5
     READ 101, ((C(I,J),I=1,NR),J=1,NR)
101  FORMAT (5F8.4)
     CALL TRIANG
     DO 111 J=1,100
     CALL MULTN(Z)
     DO 111 K=1,5
111  ARRAY(J,K)=Z(K)
     STOP
     END
```

For a three by three matrix subroutine TRIANG takes 10600 microseconds, and each vector is produced in 9520 microseconds.

It is noted that since the programs are written in FORTRAN, they are not in any sense optimal.

```
    SUBROUTINE TRIANG
    COMMON/NR/NR/C/C(3,3)/P/P(3,3)
    NC=1
    DQ 5 I = 1, NR
    DQ 5 J = 1, NR
5   P(I,J) = 0.0
    P(1,1)=SQRTF(C(1,1))
    IF(P(1,1).LE..0001)771,9
9   DO 15 K=2,NR
15  P(K,1)=C(K,1)/P(1,1)
18  DO 83 JH=2,NR
    Q=0.0
    NC=NC+1
    NCM=NC-1
    NCP=NC+1
    DO 50 J=1,NCM
50  Q=Q+P(NC,J)*P(NC,J)
    X=C(JH,JH)-Q
    IF (X.LE..000005) 777,505
505 P(JH,JH)=SQRTF(X)
51  DO 55 K=NCP,NR
    S=0.0
    DO 53 JQ=1,NCM
53  S=S+P(K,JQ)*P(NC,JQ)
55  P(K,NC)=(C(K,NC)-S)/P(NC,NC)
83  CONTINUE
    GO TO 666
771 DO 773 L=1,NR
773 P(L,1)=0.0
    GO TO 18
777 DO 780 L=JH,NR
780 P(L,JH)=0.0
    GO TO 83
666 CONTINUE
    RETURN
    END
```

```
      SUBROUTINE MULTN(Z)
      COMMON/NR/NR/P/P(3,3)
      DIMENSION Z(10),RZM(10)
      DO 220 J=1, NR
      Z(J)=0.0
  220 RZM(J)=ZMARS(DUM)
      DO 230 L=1,NR
      DO 230 M=1,NR
  230 Z(L)=Z(L)+P(L,M)*RZM(M)
      RETURN
      END
```

Appendix VIII

The tests are for a sample size of 100 vectors.

The first matrix tested was a 10 by 10 identity matrix. The maximum likelihood estimate of the covariance matrix was:

$$\begin{bmatrix}
.814 & -.048 & .049 & .127 & .085 & .001 & .073 & -.110 & -.129 & -.081 \\
-.048 & 1.112 & .133 & .079 & .245 & -.040 & .152 & .017 & -.065 & -.011 \\
.049 & .133 & .836 & .133 & -.017 & .035 & .071 & -.023 & -.131 & -.185 \\
.127 & .079 & .133 & 1.032 & .074 & -.027 & .114 & -.067 & -.090 & -.270 \\
.085 & .245 & -.017 & .074 & .972 & -.030 & -.034 & -.036 & .108 & .096 \\
.001 & -.040 & .035 & -.027 & -.030 & 1.059 & -.021 & -.034 & -.085 & -.025 \\
.073 & .152 & .071 & .114 & -.034 & -.021 & .970 & .001 & -.136 & -.077 \\
-.110 & .017 & -.023 & -.067 & -.036 & -.034 & .001 & .966 & .147 & .045 \\
-.129 & -.065 & -.131 & -.090 & .108 & -.085 & -.136 & .147 & 1.047 & -.009 \\
-.081 & -.011 & -.185 & -.270 & .096 & -.025 & -.077 & .045 & -.009 & 1.075
\end{bmatrix}$$

Covariance Matrix Input	Triangularized Matrix	Covariance Matrix Estimate
$\begin{bmatrix} 2 & 1 & 2 \\ 1 & 3 & 3 \\ 2 & 3 & 4 \end{bmatrix}$	$\begin{bmatrix} 1.414 & 0 & 0 \\ .707 & 1.581 & 0 \\ 1.414 & 1.265 & .632 \end{bmatrix}$	$\begin{bmatrix} 2.227 & 1.154 & 2.318 \\ 1.154 & 3.006 & 3.066 \\ 2.318 & 3.066 & 4.245 \end{bmatrix}$
$\begin{bmatrix} 2 & 1 & 2 & 3 \\ 1 & 3 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 7 \end{bmatrix}$	$\begin{bmatrix} 1.414 & 0 & 0 & 0 \\ .707 & 1.581 & 0 & 0 \\ 1.414 & 1.265 & .732 & 0 \\ 2.121 & 1.581 & .000 & 0 \end{bmatrix}$	$\begin{bmatrix} 1.69 & .79 & 1.62 & 2.48 \\ .79 & 3.28 & 3.16 & 4.07 \\ 1.62 & 3.16 & 3.91 & 4.78 \\ 2.48 & 4.07 & 4.78 & 6.55 \end{bmatrix}$
$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & -1 & -2 \\ 0 & -1 & 3 & 1 \\ 0 & -2 & 1 & 4 \end{bmatrix}$	$\begin{bmatrix} 1.000 & 0 & 0 & 0 \\ 0 & 1.414 & 0 & 0 \\ 0 & -.707 & 1.581 & 0 \\ 0 & -1.414 & .000 & 1.414 \end{bmatrix}$	$\begin{bmatrix} .848 & -.038 & -.031 & .064 \\ -.038 & 2.335 & -.998 & -2.673 \\ -.031 & -.998 & 2.769 & 1.138 \\ .064 & -2.673 & 1.138 & 5.169 \end{bmatrix}$
$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & -1 & -2 \\ 0 & -1 & 3 & 1 \\ 0 & -2 & 1 & 4 \end{bmatrix}$	$\begin{bmatrix} 1.000 & 0 & 0 & 0 \\ 0 & 1.414 & 0 & 0 \\ 0 & -.707 & 1.581 & 0 \\ 0 & -1.414 & .000 & 1.414 \end{bmatrix}$	$\begin{bmatrix} .950 & .014 & -.272 & -.008 \\ .014 & 2.202 & -.775 & -2.155 \\ -.272 & -.775 & 2.501 & .614 \\ -.008 & -2.155 & .614 & 3.972 \end{bmatrix}$

Covariance Matrix Input	Triangularized Matrix	Covariance Matrix Estimate
$\begin{bmatrix} 5 & 1 & 2 \\ 1 & 2 & 1 \\ 2 & 1 & 4 \end{bmatrix}$	$\begin{bmatrix} 2.236 & 0 & 0 \\ .447 & 1.673 & 0 \\ .894 & .359 & 1.752 \end{bmatrix}$	$\begin{bmatrix} 5.568 & 1.181 & 2.498 \\ 1.181 & 2.948 & .926 \\ 2.498 & .926 & 4.164 \end{bmatrix}$
		$\begin{bmatrix} 4.282 & 1.221 & 1.512 \\ 1.221 & 2.882 & 1.119 \\ 1.512 & 1.119 & 4.612 \end{bmatrix}$
		$\begin{bmatrix} 3.735 & .729 & 1.694 \\ .729 & 3.070 & 1.471 \\ 1.694 & 1.471 & 3.680 \end{bmatrix}$
		$\begin{bmatrix} 7.481 & 1.169 & 2.773 \\ 1.169 & 3.172 & .656 \\ 2.773 & .656 & 4.046 \end{bmatrix}$
		$\begin{bmatrix} 4.663 & 1.006 & 1.777 \\ 1.006 & 3.034 & 1.212 \\ 1.777 & 1.212 & 3.647 \end{bmatrix}$

Appendix IX.

The Poisson distributed random number generator.

The numbers are generated by an initial use of the variable 'NPOISSET(mean)', which initializes the generator for the desired value of the mean. This is followed by calls to 'NPOIS(DUM)' to actually produce the random numbers. The argument 'DUM' is not used. For example for a desired mean of 3.0:

```
X(1) = NPOISSET(3.0)
.
.
X(J) = NPOIS(DUM)
```

External calls are made to EXPF.

The observed average generation time per number was found to be approximately representable by: TIME = 600 + 630(MEAN).

```
          IDENT     NPOIS
          ENTRY     NPOISSET,NPOIS
R         OCT       17777777777777777
UNIFORM   SLJ       **
          ENQ       0
          LDA       R
          LLS       19
          SCL       =04000000000000000
          ADD       R
          ADD       R
          ADD       R
          STA       R
          ARS       11
          ADD       =02000000000000000
          FAD       =02000000000000000
          SLJ       UNIFORM
NPOISSET  SLJ       **
          LDA       *
          ALS       +24
          SAU       *+2
+         INA       +1
          SAU       EXIT+1
+         LDA       **
```

```
            ALS       +24
            SAU       *+1
+           LDA       **
            SCM       =077777777777777
            CALL      EXPF
+           STA       =ST1
            SLJ       START
NPOIS       SLJ       **
            LDA       *
            ALS       +24
            INA       +1
            SAU       EXIT+1
START       SIL     1 EXIT
            ENI     1 0
            RTJ       UNIFORM
+           THS       T1
            SLJ       A1
            SLJ       EXIT
A1          STA       =SY
            RTJ       UNIFORM
            INI     1 +1
            FMU       Y
            THS       T1
            SLJ       A1
EXIT        ENA     1 0
            ENI     1 **
            SLJ       **
            END
```

Appendix X.

Test results for the Poisson generator.

The test consists of ten consecutive samples of either 1000 or 5000 numbers each-for various parameter values.

PARAMETER	M1	M2	M3	M4	d/n	CHIX	SAMPLE SIZE
0.5	.5010	.4965	.4713	1.1076			1000
	.5150	.5383	.5993	1.6819			
	.4790	.4720	.4909	1.2574			
	.5070	.5105	.5387	1.4413			
	.4950	.5125	.5571	1.4485			
	.4960	.4785	.4080	.8790			
	.4800	.4641	.4171	.9417			
	.4620	.4911	.5190	1.1739			
	.4860	.4683	.4465	1.0879			
	.5010	.5105	.5223	1.3104			
theor.	.5	.5	.5	1.25			
1.0	.9850	.9998	.9388	3.5138	.0131	2.1235	1000
	.9940	1.0690	1,1868	4.7322	.0181	4.7010	
	.9790	.9755	1.0516	4.2285	.0123	4.7794	
	1.0250	.9613	.8375	3.3099	.0209	4.9883	
	.9990	.9880	.9408	3.7028	.0058	2.1281	
	.9610	.9004	.7044	2.7304	.0193	9.8119	
	1.0030	.9539	.8249	3.2231	.0079	2.3483	
	.9740	.9523	.8548	3.5808	.0161	14.0873	
	.9900	.9769	.9100	3.3157	.0152	7.1024	
	1.0310	1.0571	1.1061	4.4782	.0117	2.3473	
theor.	1.0	1.0	1.0	4.0	.0386(10%)	11.651(10%)	
2.0	1.9590	2.0994	2.5166	17.4173	.0193		1000
	1.9690	2.0120	2.3470	15.9226	.0143		
	2.0540	2.0031	1.6185	11.7484	.0200		
	1.9610	1.7893	1.2660	9.2416	.0109		
	2.0190	2.0687	2.2118	16.0789	.0130		
	1.9610	1.9374	2.1740	13.7535	.0220		
	1.9840	1.8996	1.8807	12.3351	.0213		
	1.9220	1.8618	1.5925	10.5854	.0290		
	2.0120	2.2381	2.6755	18.2820	.0147		
	1.9990	2.0110	2.0572	13.8852	.0073		
theor.	2.0	2.0	2.0	14.0	.0386(10%)		

PARAMETER	M1	M2	M3	M4	d/n	Sample Size	Generation Time (microsecs)
4.0	3.9578	3.8668	3.5334	47.2482	.013	5000	3128
	4.0214	4.1002	4.1851	54.9297	.008		
	4.0204	4.0684	3.6521	51.6736	.012		
	4.0422	4.0116	4.0824	51.6602	.010		
	3.9770	4.1349	4.6342	59.4956	.010		
	3.9880	4.0679	4.4847	58.0892	.006		
	3.9920	4.0664	4.7496	58.4535	.005		
	4.0130	3.8916	3.1439	46.6069	.014		
	4.0318	4.0536	4.1740	53.8986	.008		
	3.9806	3.9426	3.9144	50.7897	.005		
Theor.	4.0	4.0	4.0	52.0	.017 (10%)		
5.0	4.9340	4.9742	5.3430	78.8045	.016	1000	3735
	5.0462	5.0975	4.9371	80.2735	.014		
	5.0108	4.9345	4.6862	74.9158	.005		
	4.9912	5.0533	5.1257	80.9384	.004		
	5.0038	5.0412	5.7602	87.0035	.003		
	5.0126	5.0367	5.3394	83.8973	.005		
	5.0346	4.9844	4.8206	80.1724	.010		
	5.0016	5.0142	5.1778	77.7648	.006		
	5.0214	4.9419	4.7638	75.7139	.006		
	4.9598	5.0140	4.8610	77.9824	.009		
Theor.	5.0	5.0	5.0	80.0	.039 (10%)		
10.0	9.9760	9.6751	10.0922	276.3385	.013	1000	6935
	9.8280	9.5560	9.8887	280.6076	.030		
	9.9070	10.4328	12.0257	322.7622	.027		
	10.1540	9.8441	9.4111	287.9662	.019		
	10.1090	10.8179	10.5374	347.2039	.021		
	9.9710	9.9922	8.4548	305.9784	.012		
	10.0520	10.2676	11.4099	298.6314	.029		
	10.1500	10.3258	11.2464	307.3533	.020		
	9.9940	9.4594	7.1034	265.9739	.017		
	10.0340	9.6445	8.7856	283.3459	.011		
Theor.	10.0	10.0	10.0	310.0	.39 (10%)		

PARAMETER	M1	M2	M3	M4	d/n	Sample Size
5.0	4.9340	4.9742	5.3430	78.8045	.016	5000
	5.0462	5.0975	4.9371	80.2735	.014	
	5.0108	4.9345	4.6862	74.9158	.005	
	4.9912	5.0533	5.1257	80.9384	.004	
	5.0038	5.0412	5.7602	87.0035	.003	
	5.0126	5.0367	5.3394	83.8973	.005	
	5.0346	4.9844	4.8206	80.1724	.010	
	5.0016	5.0142	5.1778	77.7648	.006	
	5.0214	4.9419	4.7638	75.7139	.006	
	4.9598	5.0140	4.8610	77.9824	.009	
Theor.	5.0	5.0	5.0	80.0	.017	(10%)
	9.9948	10.0720	10.4326	303.7092	.006	5000
	10.0402	9.9338	9.4176	292.8100	.011	
	10.0174	10.1211	12.3246	335.7205	.004	
	10.0260	9.8721	9.3898	296.6628	.016	
	10.0852	9.8547	9.8097	302.7969	.017	
	10.0176	9.8136	9.8867	294.7293	.011	
	9.9914	10.1918	11.8066	323.7264	.006	
Theor.	10.0	10.0	10.0	310.0	.017	(10%)

Appendix XI.

Exponential random number generator.

The numbers are generated by a call to 'EXPRN(DUM)'. The argument 'DUM' is not used. For example: Y = EXPRN(DUM) + 4.2

No external calls are made. The observed average generation time for one number is 2270 microseconds. A conversion for numbers with parameter other than one is given on page 31 in Section 8.1.

```
          IDENT     EXPRN
          ENTRY     EXPRN
P         DEC       1.00000000
          DEC       .99999984
          DEC       .99999824
          DEC       .99998381
          DEC       .99986834
          DEC       .99906004
          DEC       .99421023
          DEC       .96996120
          DEC       .87296508
          DEC       .58197672
Q         DEC       1.00000000
          DEC       .99999989
          DEC       .99999970
          DEC       .99999917
          DEC       .99999774
          DEC       .99999386
          DEC       .99998330
          DEC       .99995460
          DEC       .99987659
          DEC       .99966454
          DEC       .99908812
          DEC       .99752125
          DEC       .99326205
          DEC       .98168436
          DEC       .95021293
          DEC       .86466471
          DEC       .63212055
TABLE     BSS       20
MIN       DEC       0.0
R         OCT       540203450727422
UNIFORM   SLJ       **
          ENQ       0
          LDA       R
          LLS       19
```

[illegible]

```
         SCL      =0400000000000000
         ADD      R
         ADD      R
         ADD      R
         STA      R
         ARS      11
         ADD      =0200000000000000
         FAD      =0200000000000000
         SLJ      UNIFORM
EXPRN    SLJ      **
         LDA      *
         ALS      24
         INA      1
         SAL      A4+1
         SIL    1 A4
         SIU    2 A4+1
         ENI    1 +10
         RTJ      UNIFORM
+        THS    1 P
         SLJ      A4
+        ENI    2 -1
A1       INI    2 +1
         RTJ      UNIFORM
+        STA    2 TABLE
+        ISK    1 +9
         SLJ      A1
+        STA      MIN
A2       IJP    2 A31
         SLJ      A3
A31      LDA    2 TABLE
         THS      MIN
         SLJ      A2
         STA      MIN
         SLJ      A2
A3       LDA      MIN
+        ENI    1 +17
         RTJ      UNIFORM
+        THS    1 Q
         SLJ      A4
         ENA    1 0
         INA      -16
         SCM      =0777777777777777
         ADD      =0204400000000000
         FAD      =0204400000000000
A4       FAD      MIN
         ENI    1 **
         ENI    2 **
         SLJ      **
         END
```

Appendix XII.

Test results for the exponential generator.

The test consists of consecutive samples of 1000 each for lambda equal one.

SAMPLE	M1	M2	M3	M4	d/n	Generation Time
1000-1	1.0056	.9544	1.5731	6.0113	.015	
-2	1.0602	1.1981	2.9340	15.0479	.020	
-3	1.0606	1.2050	2.9526	15.5327	.037	
-4	1.0272	1.1211	2.5961	12.1087	.020	
-5	.9422	.8821	1.7657	7.6101	.030	
-6	1.0244	.9484	1.4382	5.2094	.031	
-7	.9273	.8457	1.4637	5.7615	.038	
-8	1.0520	1.1072	2.4988	12.2010	.033	
-9	1.0323	1.2839	3.8978	23.6524	.016	
-10	.9751	.8368	1.3691	5.3359	.022	2270 microsecs
Theor.	1.0	1.0	2.0	9.0	.039 (10%)	
-11	.9854	1.0562	2.4394	11.6124	.018	
-12	.9998	1.0495	2.2430	10.3795	.014	
-13	.9841	1.0447	2.2683	10.1315	.037	
-14	1.0204	1.0517	1.9077	7.1153	.021	
-15	.9513	.7981	1.1864	4.1650	.022	
-16	.9592	.9722	1.7937	6.9373	.041	
-17	1.0403	1.1069	2.2454	9.9354	.017	
-18	1.0329	1.0635	1.9497	7.6527	.018	
-19	.9711	.9014	1.6697	6.8324	.022	
-20	.9835	.9451	1.7445	7.1993	.020	
-21	1.0051	1.0281	2.1547	9.4030	.015	
-22	.9631	.8759	1.3956	4.9154	.020	
-23	.9951	.9813	1.7784	7.1579	.015	
-24	1.0176	1.1237	2.5168	12.1998	.020	
-25	1.0278	1.0664	2.2520	10.2068	.025	
-26	1.0317	1.0396	2.1303	9.3136	.027	
-27	1.0188	.9595	1.6659	6.7573	.024	
-28	.9334	.8338	1.3261	4.7659	.038	
-29	.9468	.9008	1.8411	8.4889	.034	
-30	1.0172	.9759	1.8041	7.5746	.029	

Addendum 1.

Results of the investigation of the tails of the Marsaglia normal random number generator.

The test results belie the idea resulting from investigations made in Section 4.3. A typical series of results of the tests are shown below. The first line gives the theoretical cumulative sample result based on a sample size of 100. The following data shows the experimental results. The first interval is from minus infinity to -2.56, following intervals are 0.16 in width. Thus only the negative half of the distribution is shown.

.52	.82	1.25	1.88	2.74	3.92	5.48	7.49	10.03	13.14	16.85	21.19	26.11
1	1	3	3	3	5	7	8	10	11	15	17	27
0	0	0	0	1	1	1	1	1	5	11	15	18
1	1	1	1	2	2	3	4	8	12	17	20	23
0	1	1	4	6	10	12	13	15	16	19	26	33
0	1	2	4	5	6	8	10	12	16	22	26	29
1	1	1	3	4	7	9	10	13	14	16	21	28
0	0	0	1	1	1	3	7	11	13	18	21	25
0	0	1	2	3	5	6	8	10	13	17	19	25
0	0	3	3	4	6	9	12	12	14	18	26	30
1	1	3	5	7	8	8	8	8	11	17	21	26